36장 디스트럭처링 할당

구조 분해 할당(destructuring assignment)은 구조화된 배열과 같은 이터러블 또는 객체를 destructuring(비구조화, 구조 파괴) 하여 1개 이상의 변수에 개별적으로 할당하는 것

36.1 배열 디스트럭처링 할당

ES6 배열 디스트럭처링 할당은 배열의 각 요소를 배열로 부터 추출하여 1개 이상의 변수에 할당하는데, **배열 디스트럭처링 할당의 대상(할당문의 우변)은 이터러블이어야 하며, 할당 기준은 배열의 인덱스**이다.

```
const arr = [1, 2, 3];

const [one, two, three] = arr;

console.log(one, two, three); // 1 2 3

const [four, five] = {}; // TypeError
```

배열 구조분해 할당을 위한 변수에 기본값을 설정할 수 있다.

단, 기본값보다 할당된 값이 우선한다.

```
const [e, f = 10, g = 3] = [1, 2];
console.log(e, f, g); // 1 2 3
```

배열 구조분해 할당을 위한 변수에 Rest 파라미터와 유사하게 Rest 요소 ...를 사용할 수 있다.

```
const [x, ...y] = [1, 2, 3];
console.log(x, y); // 1 [ 2, 3]
```

36.2 객체 디스트럭처링 할당

- ES6의 객체 구조분해 할당은 객체의 각 프로퍼티를 객체로부터 추출하여 1개 이상의 변수에 할당
    - 이때 객체 구조분해 할당의 대상은 객체이며, 할당 기준은 프로퍼티 키

    - 즉, 순서는 의미가 없고 선언된 변수 이름과 프로퍼티 키가 일치하면 할당

```
const user = { firstName: 'Yongwoo', lastName: 'Cho' };

const { lastName, firstName } = user;
console.log(firstName, lastName); // Yongwoo Cho
```

- 할당 연산자 왼쪽에 프로퍼티 값을 할당받을 변수를 선언해야 하는데 이때 변수를 객체 리터럴로 선언한다.
- 우변에 객체 또는 객체로 평가될 수 있는 표현식(문자열, 숫자, 배열 등)을 할당하지 않으면 에러가 발생한다.
- 객체 구조분해 할당을 위한 변수에 기본값을 설정할 수 있다.

```
const { firstName = 'Yongwoo', lastName } = { lastName: 'Ch
o' };

console.log(firstName, lastName); // Cho Yongwoo
```

객체 구조분해 할당은 객체에서 프로퍼티 키로 필요한 프로퍼티 값만을 추출하여 변수에 할당하고 싶을 때 유용하다.

```
const todo = { id: 1, content: 'HTML', completed: true};
// todo 객체로부터 id 프로퍼티만 추출한다.
const { id } = todo;
console.log(id); // 1
```

객체를 인수로 전달받는 함수의 매개변수에도 사용할 수 있다.

```
function printTodo(todo) {
  console.log(
    `할일 ${todo.content}은 ${todo.completed ? '완료' : '비완
료'} 상태입니다.`);
}

function printTodo({ content, completed }) {
  console.log(`할일 ${content}은 ${completed ? '완료' : '비완
```

```
료'} 상태입니다.`);
}
```

배열의 요소가 객체인 경우 배열 구조분해 할당과 객체 구조 분해할당을 혼용할 수 있다.

```
const todos = [
  { id: 1, content: 'HTML', completed: true },
  { id: 2, content: 'CSS', completed: false },
  { id: 3, content: 'JS', completed: false }
];
const [, { id }] = todos;
console.log(id); // 2
```

- 객체 구조분해할당을 위한 변수에 Rest 프로퍼티 ... 를 사용할 수 있다.
- Rest 프로퍼티는 Rest 파라미터나 Rest 요소와 마찬가지로 반드시 마지막에 위치해야 한다.

```
const { x, ...rest } = { x: 1, y: 2, z: 3 };

console.log(x, rest); // 1 {y:2, z:3}
```